La simulation numérique : Traitement d'images

Série d'exercices N° 1

Dans la suite des exercices, utiliser le module PIL pour manipuler les images en Python.

```
>>> from PIL.Image import *  #Voir annexe pour plus d'informations
```

Exercice 1.
Écrire un script Python permettant d'inverser les niveaux de gris d'une image .

Exercice 2.
Écrire un script Python permettant d'ajouter autour d'une image une bordure d'épaisseur **e** pixels. Attention, cette bordure vient se placer autour de l'image, elle n'écrase pas les pixels existants.

Exercice 3.
Écrire un script Python permettant de **c**réer une image carrée (par exemple de taille 100 sur 100) ayant la forme d'un X noir sur fond blanc joignant les coins du carrée.

Exercice 4.
Écrire un script Python permettant d'ouvrir une image et compter le nombre de pixels parfaitement noirs et le nombre de pixels parfaitement blancs dans celle-ci.

Exercice 5.
Écrire un script Python permettant d'ouvrir une image et d'enregistrer une copie agrandie d'un facteur 2, en hauteur comme en largeur. Chaque pixel de l'image originelle est donc remplacé par quatre pixels identiques.

Exercice 6.
Écrire un script Python permettant d'ouvrir une image et d'afficher l'image symétrique par rapport à un axe vertical. Même question en faisant subir cette fois à l'image une rotation de 90°.

Exercice 7.
Écrire une fonction Python qui prend comme arguments deux entiers **n** et **p** et crée l'image d'un damier de dimensions n × p : les pixels sont alternativement noirs et blancs le long des lignes comme le long des colonnes.

Annexe

Fonction	Signification

`from PIL.Image import *`
`import PIL.Image`
`import PIL.Image as im`
	PIL(Python Imaging Library)
	PIL est un module de traitement d'images

`tiger = open('tiger.jpg')`	Ouvrir un fichier image et créer un objet Image
`tiger.show ()`	Afficher l'image dans une fenêtre

`data = tiger.getdata()`
`L = list(data)`
`print(L)`
	Retourner une séquence de valeurs des pixels d'une image

`Largeur, hauteur = tiger.size` `print(Largeur, hauteur)`	Récupérer la largeur(nombre de colonnes) et la hauteur (nombre de lignes) de l'image
`Img_gris = tiger.convert('L')`	Retourner une copie d'une image convertie dans un mode différent. Les modes d'images en Python : ✔ « 1 » : Mode 1 bit ou mode Noir et Blanc ✔ « L » : Mode 8 bits ou mode Niveaux de gris ✔ « RGB » : Mode 24 bits ou mode RVB
`Img_gris.save('tigergris.bmp')`	Sauvegarder une image dans un fichier
`n=new('RGB',(Largeur,hauteur))`	Création d'un nouvel objet Image en précisant le mode et les dimensions Par défaut, tous les pixels sont mis à 0. l'image est noire.

`new('RGB',(100,100),(123,34,5))`
`new('RGB',(100,100),"red")`
`new('L',(100,100),123);`
`new('1',(100,100),"white")`
	`créer une image avec une couleur par défaut`

`nouveau.putdata(list(data))`	Mettre à jour les codes des pixels d'une image
`p = Img_gris.getpixel((0,1))`	Récupérer la valeur d'un pixel à une position donnée
`Img_gris.putpixel((0,1),255)`	Modifier la valeur d'un pixel à une position donnée
`tiger1 = tiger.copy ()`	Créer une copie conforme d'une image
`tiger1.resize((120,120)).show()`	Créer une copie redimensionnée d'une image
`tiger.transpose(FLIP_LEFT_RIGHT).show()`	Créer une copie transposée d'une image
`Img_gris.rotate(45).show()`	Créer une copie de l'image originale déviée d'un angle de 45 degrés à partir de son centre dans le sens contraire des aiguilles d'une montre.